УДК 004.42 **doi: 10.26906/SUNZ.2018.4.125**

В. Ю. Мерлак [1], І. С. Зиков [2], Г. І. Молчанов [2]

[1] Національний аерокосмічний університет імені М. Є. Жуковського «ХАІ», Харків, Україна,
[2] Національний технічний університет "Харківський політехнічний інститут", Харків, Україна

СИТУАЦІЙНО-ОРІЄНТОВАНИЙ ПІДХІД ПРИ ПРОЕКТУВАННІ ВІДЖЕТІВ

Предмет статі – проектування веб-застосунків, котрі забезпечують застосування технологій критичного комп'ютингу, зокрема, можливостям безперебійної роботи, високої доступності та резервного копіювання. **Метою даною статі** є проведення аналізу можливості використання ситуаційно-орієнтованого підходу при проектуванні віджетів. **Результати роботи.** Виконано дослідження застосування віджетів у сучасних технологіях. Обрано основні характеристики якості з точки зору впливу на архітектуру системи. Виділено основні архітектури, які використовуються для взаємодії з користувачем, визначені критерії порівняння і результати були приведені в таблиці, виходячи з якої видно, що віджет задовольняє характеристики якості з точки зору впливу на архітектуру системи. **Висновки**. Проблема оптимізації доступу до даних за часом особливо істотна для ієрархічних реляційних структур даних, однак при такій організації даних існує багато математичних методів, що дозволяють прискорити доступ до даних в конкретних випадках. Одним із найбільш прийнятних підходів є використання при проектуванні віджетів ситуаційно-орієнтованого апарату, що забезпечить виконання вимог критичного комп'ютингу. Ключовими перевагами такого підходу є такі: можливість взаємодії з даними в контексті ситуації, розвиток якої описується за допомогою динамічної моделі кінцевих станів (вихідної моделі); можливість наочного представлення процесу; автоматизація проектування веб-застосунки за рахунок збільшення рівня абстракції при побудові моделі програми та виконання тривіальних функцій інтерпретатором моделі.

Ключові слова: віджет, веб-застосунок, критичний комп'ютинг, ситуаційно-орієнтований підхід.

Вступ

Для початку треба визначити, що таке "віджет" і який сенс вкладено у статті в це визначення. Технологія «віджетів» досить-таки нова технологія та її сутність досить проста, але має величезну важливість і популярність при розвитку сучасних стартапів та проектів. За допомогою цієї технології можливо вбудувати функціонал своєї системи на будь-який сайт за допомогою буквально одного рядка коду. Віджет - це невеликий незалежний програмний модуль, який працює в деякому середовищі (наприклад, на сайті, в браузері, у мобільному телефоні) і виконує, як правило, одну певну функцію [1]. Їх можна розділити на групи по середовищам, в яких вони працюють. Прийнято виділяти два види: веб-віджети і віджети для робочого столу.

Віджети для рабочого столу - це невеликі інструменти (програми), що виконують якусь одну функцію і вимагають для своєї роботи спеціальної середовищі - віджет-движка [2]. Приклад можна побачити на рис. 1.

Графічний спосіб подачі інформації для користувача є більш переважним. Саме для цього зручно використовувати інтерактивні елементи сайту, які мотивують користувача до здійснення дії - веб-віджети (рис. 2) – змінні компоненти користувальницького інтерфейсу веб-сторінки, що дозволяють користувачеві взаємодіяти з сайтом.

Аналіз літератури. Методам розробки веб-застосунків на сьогодні присвячено багато теоретичних розробок, зокрема, наведені у джерелах [1-5].

Але врахуванню застосування технологій критичного комп'ютингу у даних роботах привчено мало уваги, зокрема, можливостям безперебійної роботи, високої доступності та резервного копіювання. У багатьох випадках даним умовам відповідають віджети.

Тому **метою даною статі** є проведення аналізу можливості використання ситуаційно-орієнтованого підходу при проектуванні віджетів.

Рис. 1. Приклад віджету для робочого столу

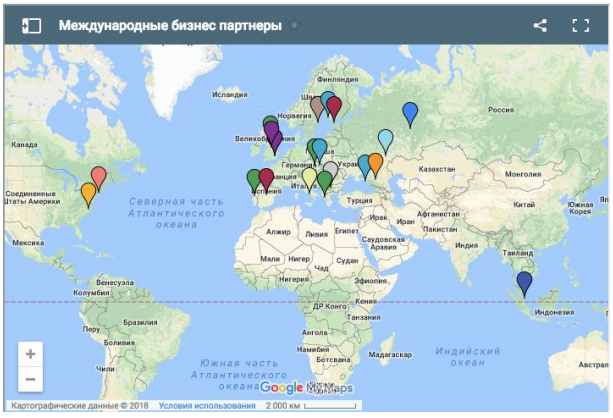

Рис. 2. Приклад веб-віджету.

© Мерлак В. Ю., Зиков І. С., Молчанов Г. І., 2018

1. Порівняння архітектур взаємодії віджету з користувачем

Компанія Google використовує віджети для розробки свого програмного забезпечення для веб-застосунків і мобільних пристроїв [3]. У продуктах компанії віджети забезпечують виконання таких дій: структура для інших віджетів (карти і розділи), інформація для користувача (текст і зображення), доступність для дій (кнопки, поля введення тексту або прапорці).

На підставі статті [4] було обрано такі характеристики якості з точки зору впливу на архітектуру системи:

- вимоги до повторного використання реалізації або компонентів програми або системи (Reusability). Найчастіше ці вимоги будуть виникати там, де загальні компоненти використовуються декількома модулями розробляється вами системи;

- вимоги до розширюваності (Extensibility) застосунки або системи в зв'язку з появою нових функціональних вимог, тісно пов'язане з таким архітектурним атрибутом якості, як переносимість коду;

- вимоги до переносимості (Portability) застосунки або системи на інші платформи;

- вимоги до взаємодії між компонентами рішення, між зовнішніми компонентами, використання стандартних протоколів і технологій взаємодії (Interoperability); до таких вимог можна віднести можливість використання декількох стандартних протоколів для обміну даними між однією з підсистем розроблюваної системи і зовнішньою системою-постачальником даних;

- вимоги до підтримки системи або застосунки (Supportability); серед цих параметрів можуть бути названі такі як, наприклад, дешевизна і швидкість розробки, прозорість поведінки застосунки, простота аналізу помилок і проблем в роботі;

- вимоги до модульності програми або системи (Modularity). Зазвичай такі вимоги вказують, яким чином система повинна бути розділена на модулі, або перераховують список обов'язкових модулів, які повинні входити до складу системи;

- вимоги до можливості тестування (Testability) застосунки або системи визначають обсяг вимог до автоматичного і ручного тестування, наявність необхідного інструментарію;

- вимоги до можливості і простоті локалізації (Localizability) застосунки або системи визначають можливості і специфічні архітектурні вимоги, що накладаються процесом локалізації.

Розглянемо підходи до створення веб-застосунків. Прийнято виділяти шаблонний і ручної підходи до створення веб-сторінок. Шаблонний метод - поведінковий шаблон проектування, що визначає основу алгоритму і дозволяє спадкоємцям перевизначити деякі кроки алгоритму, не змінюючи його структуру в цілому [5].

І ручний, тобто верстка веб-сторінок - створення структури html-коду, яка розміщує елементи веб-сторінки (зображення, текст і т. д.) у вікні браузера, згідно з розробленим макету, таким чином, щоб елементи дизайну виглядали аналогічно макету.

Проаналізувавши інформацію, були виділені основні архітектури, які використовуються для взаємодії з користувачем і визначені критерії порівняння. Результати порівняння наведені в табл. 1.

На основі табл. 1, приведеної нижче, видно, що віджет задовольняє характеристики якості з точки зору впливу на архітектуру системи.

Таблиця 1 – **Порівняння архітектур побудови призначених для користувача інтерфейсів для веб-застосунків**

Архітектура / Критерій	Отримання сторінок з боку сервера	Односторінкові застосунки	Віджети
Місце формування інтерфейсу	Сервер	Клієнт (браузер)	Клієнт (браузер)
Реакція сервера у відповідь на отримання запиту про зміну даних	Генерація та відправка сторінки	Відправка JSON	Відправка JSON
Пошукова оптимізація (SEO)	+	-	-
Швидкодія	-	+	+
Адаптивність дизайну	-	+	+
Економія трафіку	-	+	+
Можливість підтримки взаємодії з користувачем при втраті з'єднання з мережею Internet	-	+	+
Можливість вбудовування стороннього сайту	-	-	+
Використовуваний інструментарій	ASP.NET (C#), Thymeleaf (Java), JSP (Java), JSF (Java)	ReactJS, Angular, Vue.js, Ember.js, Backbone.js	

Однак в джерелі [6] помічено, що доводиться витрачати більше часу на створення тестів для інтерфейсу, так як користувач може змінювати стан через послідовності взаємодій, і багато помилок можуть не відображатися до тих пір, поки не відбудеться певна взаємодія. Тому цей пункт теж потрібно враховувати.

У даній роботі увагу більше направлено на ієрархічні веб-віджети. Концепція ієрархічних віджетів полягає в тому, що в станах динамічної моделі передбачаються віджет-елементи, відповідні фрагментами зображення на екрані користувача і задають спосіб і параметри формування результуючого HTML- коду, а також посилання на батьківські віджети, які об'єднують віджет-елементи в ієрархію. У процесі інтерпретації динамічної моделі в залежності від її поточного стану автоматично формується результуючий контент ієрархії віджет-елементів, який по завершенні виводиться у вихідний інформаційний потік [7]. Завдання ієрархічних віджетів при взаємодії з користувачем: сформувати HTML-код, що задає структуру і зміст зображення на екрані, який пересилається в

браузер користувача разом з CSS-кодом, що задає параметри зовнішнього вигляду зображення, і JavaScript кодом, що задає активність на стороні клієнта; прийняти дані, введені користувачем, перевірити їх коректність, перетворити, зберегти в серверній базі даних, змінити поточний стан для переходу до нового циклу взаємодії з користувачем.

Особливо актуальне використання віджетів для систем критичного застосування, в яких час обробки транзакції має бути дуже мало, тому що дані системи повинні функціонувати в режимі, близькому до режиму реального часу. Зокрема, ця вимога актуальна для віджетів, які використовуються в системах підтримки прийняття рішень при ліквідації наслідків надзвичайних ситуацій.

А клієнтський підхід до візуалізації призначеного для користувача інтерфейсу на увазі, що всі необхідні дані, такі як бізнес-логіка і шаблони, завантажуються повністю. Він робить клієнтський інтерфейс досить швидким і чуйним на дії користувача [8]. У [9] було проведено порівняння HSM-моделей без використання і з використанням віджета.

В даний час при програмної реалізації віджетів використовуються стандартні засоби розробки, які не враховують специфіку систем.

Зокрема, при розробці віджетів не проводиться оптимізація за часом доступу до даних, а, з огляду на специфіку сховищ даних і Запитальний транзакцій, час доступу можна скоротити в 2-3 рази.

Однак, при цьому істотно збільшуються фінансові та часові витрати на розробку віджетів.

3. Ситуаційно-орієнтований (модельно-орієнтований) підхід

У [6, 7] запропонований сітуаціонно-орієнтований підхід до проектування веб-застосунків, зокрема віджетів. З точки зору розробника застосунку суть даного підходу полягає в абстрагуванні від програмування функцій і вузлів до розробки динамічної моделі програми, відповідно до якої інтерпретатор динамічної моделі буде здійснювати переходи між ситуаціями. Модель можна спрощено представити у вигляді кінцевого графа, де вузли - це стану, а переходи - це можливі дії користувача.

На рис. 3 представлена мнемосхема процесу побудови роботи веб-програми на основі ситуаційно-орієнтованого підходу з точки зору розробника. Розробник, як було сказано вище, проектує ситуаційну модель бізнес-процесу. Визначаються стану і умови переходів між станами.

Рис. 1. Розробка віджету в ситуаційно-орієнтованому підході

Така концептуальна модель формується розробником і при ручному програмуванні. На цьому етапі, як правило, продумується основна логіка програми, структура бази даних. Потім концептуальна модель реалізується в процесі програмування. Для автоматизації цього процесу переходу від концептуальної моделі віджету до його програмної реалізації і для зниження трудомісткості треба вбудовувати динамічну модель в явному вигляді в серверний сценарій. Таким чином, генеруються допоміжні файли які підключається до виконуваної сторінки. Крім того, структура динамічної моделі відповідає структурі формується інтерфейсу користувача. Внаслідок цього модель є ієрархічною. Наочність моделі - це одна з переваг підходу. При створенні динамічної моделі формується база даних програми, заснована на колекції XML-документів. Інтерпретатор динамічної моделі обробляє динамічну модель, визначаючи поточний стан, і, взаємодіючи з БД, обробляє інформацію, формуючи веб-сторінку для браузера користувача. Останній в свою чергу відображає dsl;tn користувачеві і відправляє інформацію про його діях інтерпретатора, який, знову ж таки, обробляючи динамічну модель, визначає подальшу поведінку програми. Для визначення поточного стану, в якому знаходиться користувач, пропонується ввести особливий функціонал інтерпретатора - «пам'ять поточного стану». Застосування ситуаційно-орієнтованого підходу для реалізації управління віджетом повинно бути обумовлено наявністю в моделі базі даних процесів, які носять ситуативний характер. У цих умовах виникає необхідність створення і веден-

ня даних, що характеризують відповідні етапи розвитку ситуації виконання проекту.

Висновки

Одним із найбільш прийнятних підходів є використання при проектуванні віджетів ситуаційно-орієнтованого апарату, що забезпечить виконання вимог критичного комп'ютингу. Ключовими перевагами такого підходу є такі: можливість взаємодії з даними в контексті ситуації, розвиток якої описується за допомогою динамічної моделі кінцевих станів (вихідної моделі); можливість наочного представлення процесу; автоматизація проектування веб-застосунки за рахунок збільшення рівня абстракції при побудові моделі програми та виконання тривіальних функцій інтерпретатором моделі.Подальші дослідження в даному напрямку будуть спрямовані на аналіз математичних методів оптимального пошуку і розробку методу, що прискорює пошук даних для систем критичного застосування.

СПИСОК ЛІТЕРАТУРИ

1. Создание веб виджета для сторонних сайтов [Електронний ресурс]. – Режим доступу: http://resource-gsv.ru/programming/create-web-widget.html – 28.08.2018 р.
2. Что такое виджеты [Електронний ресурс]. – Режим доступу: http://widgetok.ru/2009/01/what-is-widgets/ – 28.08.2018 р.
3. Google Developers [Електронний ресурс] Widgets – Режим доступу: https://developers.google.com/gmail/addons/concepts/widgets/ – 28.08.2018 р.
4. Хабр [Електронний ресурс] Нефункциональные требования к программному обеспечению. Часть 1 – Режим доступу: https://habr.com/post/231961/ – 28.08.2018 р.
5. Википедия [Електронний ресурс] Шаблонный метод (шаблон проектирования) – Режим доступу: https://ru.wikipedia.org/wiki/Шаблонный_метод_(шаблон_проектирования) – 28.08.2018 р.
6. Blogger [Електронний ресурс] Server-side HTML vs. JS Widgets vs. Single-Page Web Apps – Режим доступу: http://blog.pamelafox.org/2013/05/frontend-architectures-server-side-html.html – 28.08.2018 р.
7. Канашин, В. В. Иерархические виджеты: алгоритмы контроля данных пользователя в веб-приложениях на основе ситуационно-ориентированных баз данных / В. В. Канашин, В. В. Миронов // Вестник УГАТУ. – Уфа, 2014. –№ 1. – С. 204-213.
8. Webix [Електронний ресурс] Client Side vs Server Side UI Rendering. Advantages and Disadvantages – Режим доступу: https://blog.webix.com/client-side-vs-server-side-ui-rendering/ – 28.08.2018 р.
9. Канашин, В. В. Иерархические виджеты: опыт применения в веб-приложении на основе ситуационно-ориентированной базы данных [Текст] / В. В. Канашин, В. В. Миронов // Вестник УГАТУ. – Уфа, 2014. –№ 2. – С. 185-196.

Рецензент: д-р техн. наук, проф. І. В. Рубан,
Харківський національний університет радіоелектроніки, Харків
Received (Надійшла) 14.06.2018
Accepted for publication (Прийнята до друку) 22.08.2018

Ситуационно-ориентированный подход при проектировании виджетов

В. Ю. Мерлак, И. С. Зыков, Г. И. Молчанов

Предмет статьи - проектирование веб-приложений, которые обеспечивают применение технологий критического компьютингу, в частности, возможностям бесперебойной работы, высокой доступности и резервного копирования. **Целью данной статьи** является проведение анализа возможности использования ситуационно-ориентова¬ного подхода при проектировании виджетов. **Результаты работы**. Выполнены исследования применения виджетов в современных технологиях. Избран основные характеристики качества с точки зрения влияния на архитектуру системы. Выделены основные архитектуры, используемых для взаимодействия с пользователем, определенные критерии сравнения и результаты были приведены в таблице, исходя из которой видно, что виджет удовлетворяет характеристики качества с точки зрения влияния на архитектуру системы**. Выводы.** Одним из самых приемлемых подходов является использование при проектировании виджетов ситуационно-ориентированного аппарата, что обеспечит выполнение требований критического компьютинга. Ключевыми преимуществами такого подхода являются: возможность взаимодействия с данными в контексте ситуации, развитие которой описывается с помощью динамической модели конечных состояний (исходной модели); возможность наглядного представления процесса; автоматизация проектирования веб-приложения за счет увеличения уровня абстракции при построении модели программы и выполнения тривиальных функций интерпретатором модели.

Ключевые слова: виджет, веб-приложение, критический компьютинг, ситуационно-ориентированный подход.

Situatio-oriented approach for designing vines

V. Merlak, I. Zykov, H. Molchanov

The subject of the article is the design of web applications that utilize critical computer technologies, such as uninterrupted work, high availability and backup. **The purpose of this article** is to analyze the possibility of using the situational-oriented approach when designing widgets. **Results of work**. The research of application of widgets in modern technologies is executed. Selected the main characteristics of quality in terms of impact on the architecture of the system. The main architectures used to interact with the user are identified, the benchmarking criteria are defined, and the results are shown in the table, based on which it is clear that the widget meets the characteristics of quality in terms of the impact on the architecture of the system. **Conclusions.** One of the most appropriate approaches is to use when designing widgets for situational-oriented devices, which will ensure that the requirements of critical computing will be fulfilled. The key benefits of this approach are: the ability to interact with the data in the context of a situation, the development of which is described by means of a dynamic model of finite states (the original model); the opportunity to visualize the process; automating the design of a web application by increasing the level of abstraction when constructing a program model and performing trivial functions as a model interpreter.

Keywords: widget, web application, critical computer, situational-oriented approach.